Kubernetes Setup

Launch 2 servers

- **Ubuntu OS**
- **Security Group (e.g. name KubernetesSG) →**
  - **22 =** ssh
  - **8080 =** alternate to http
  - **80 =** http
  - **10250 =** Kubelet API control plane uses this to talk to worker nodes
  - **6443 =** K8S API server listens for kubectl and node communication

Follow **Step 1 to Step 6** on **Both:**
Master Node and Worker nodes(on every worker node you create...)

Step 1 - Update Ubuntu

```
sudo swapoff -a
```

The command is used in Linux systems to **disable all swap space** immediately.

*→ **Swap** is extra virtual memory on your disk used when RAM is full. It's slower than RAM.*
***In Kubernetes**, we have to disable swap because it can confuse the system about available memory and cause performance or scheduling issues. K8s expects only real RAM to be used.*

```
sudo apt update
```

Updates the local package index (so APT knows about the latest available versions).

```
sudo apt upgrade
```

Upgrades all installed packages to their latest versions.

Step 2 - Install Docker

```
sudo apt install docker.io
```

Installs Docker, the container runtime that Kubernetes uses to run pods (containers).

Step 3 - Start and Enable Docker

```
sudo systemctl enable docker
```

Enables Docker to start automatically on boot.

```
sudo systemctl start docker
```
Starts the Docker service now.

```
sudo systemctl status docker
```
Shows the status of the Docker service (running, stopped, or failed).

Step 4 - Install Kubernetes Tools

```
sudo apt-get install -y apt-transport-https ca-certificates
curl gnupg
```
Installs helper tools to securely download packages over HTTPS.

Step 5 - Add Kubernetes Repository Key & Source

```
sudo mkdir /etc/apt/keyrings
```
Creates a directory to store GPG keys securely.

```
curl -fsSL
https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo
gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```
Downloads and stores the GPG key for verifying Kubernetes packages.

```
sudo chmod 644 /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```
Sets correct read permissions for the key file.

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-
keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' |
sudo tee /etc/apt/sources.list.d/kubernetes.list
```
Adds the Kubernetes official repo to your APT sources.

```
sudo chmod 644 /etc/apt/sources.list.d/kubernetes.list
```
Makes the new source file readable to the system.

Step 6 - Install Kubernetes Components

```
sudo apt-get update
```
Updates APT to include Kubernetes repo.

```
sudo apt-get install -y kubectl kubeadm kubelet
```

Installs:

- kubectl: CLI to manage Kubernetes.
- kubeadm: Tool to initialize and join clusters.
- kubelet: Agent running on every node that talks to the control plane.

Master Node Only Commands:

Step 7 - Initialize the Cluster

```
sudo kubeadm init --ignore-preflight-errors=all
```

Initializes the Kubernetes master (control plane).

(--ignore-preflight-errors=all skips setup checks — okay for testing, not recommended in production.)

Step 8 - Configure kubectl Access

```
mkdir -p $HOME/.kube
```

Makes a .kube config directory in your home folder.

```
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

Copies the admin kubeconfig file so you can run kubectl commands as your user.

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Sets file ownership to your user so you can read it without sudo.

Step 9 - Apply Network Plugin

```
kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.26.0
/manifests/calico.yaml
```

Installs Calico, a networking plugin required to enable pod-to-pod communication and network policies.

OR

```
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Docume
ntation/kube-flannel.yml
```

Installs Flannel, a networking plugin required to enable pod-to-pod communication and network policies

Step 10 - Get Join Command for Worker Nodes

```
kubeadm token create --print-join-command
```

Prints a kubeadm join command that you can run on worker nodes to connect them to this master.

After this you will see a token like this:

```
kubeadm join 172.31.19.17:6443 --token kw5ayi.d85b0vtz1wrdje1k
--discovery-token-ca-cert-hash
sha256:35f0db176895a76bdc99faad12a3ee0199af3bdfca872a0830dce1e
93ce82309
```

Worker Node Only Commands:

Step 11 - Paste token copied from master node, add sudo before

```
Example: sudo kubeadm join 172.31.19.17:6443 -token
kw5ayi.d85b0vtz1wrdje1k --discovery-token-ca-cert-hash
sha256:35f0db176895a76bdc99faad12a3ee0199af3bdfca872a0830dce1e
93ce82309
```

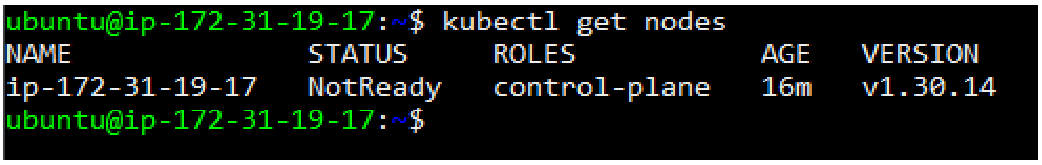
172.31.19.17 → Private ip of master node

Step 12 - To verify the connection

In Master Node run:

```
kubectl get nodes
```

If you see like this,



```
ubuntu@ip-172-31-19-17:~$ kubectl get nodes
NAME              STATUS     ROLES           AGE   VERSION
ip-172-31-19-17   NotReady   control-plane   16m   v1.30.14
ubuntu@ip-172-31-19-17:~$
```

Setup is done!!!